MySQL2WebDAV Dokumentation

I modsætningen til tidligere projekter har vi valgt ikke at bruge phpdocu til dette projekt, fordi vi kun har skrevet en klasse, med forholdsvis få medlemmer. I stedet har vi delt dokumentationen op i de funktioner som repræsentere protokol kald, og de hjælpe funktioner vi har skrevet til at behandle disse protokol kald med.

HTTP_WebDAV_Server_MySQL Dokumentation

HTTP_WebDAV_Server_MySQL nedarver fra HTTP_WebDAV_Server som implementere det meste af WebDAV protokollen.

$base
: Root direktory, lavest level i WebDAV ressourcen, i vores tilfælde "/", fordi vi ikke er baseret oven på den normale filsystem.

$db_host
: MySQL database host.

$db_name
: MySQL database navn.

$db_user
: MySQL database bruger.

$db_passwd
: MySQL database bruger password

ServeRequest()
: Servicere en anmodning, man kan godt kalde dette for en konstruktør (constructor). Da den bliver kaldt når klassen skal aktiveres. Men den er ikke en konstruktør fordi det skal være muligt at indstille database indstillingerne, før klassen forbinder til database og begynder at behandle WebDAV anmodningerne.

check_auth($type, $user, $pass)
: Verificere brugere, parametret type angiver hvilken type autentifikation der er forespurgt. Vi har lavet mindre implementering. Men den er ikke testet, da vi havde nok med debugge resten af klassen. parametret user og pass er selvfølgelig bruger navn og passwork. Denne funktion er ikke en del af WebDAV protokollen, da autentifikationen bliver behandlet i HTTP-protokollen.

Protokol kald

Disse funktioner implementere kald i WebDAV protokollen, de kaldes af base klassen og deres output parses af base klassen. Man kan goså godt betragte disse funktioner som event handlers i normalt OOP terminologi.

GET(&$options)
: Returnere en fil, eller printer et "direktory index", værdierne parses frem og tilbage gennem det reference overførte parameter *options*. Det "direktory index" som denne protokol handler implementere, er det index direktory du vil se hvis du åbne en WebDAV ressource i din HTML/Webbrowser.

MKCOL($options)

Denne funktion opretter en kollektion (mappe/katalog). Igen parses værdierne frem og tilbage gennem det reference overførte parameter *options*.

PUT(&$options)

Opretter en fil, eller opdatere en eksisterende fil. Også denne gang parses værdierne frem og tilbage gennem det reference overførte parameter *options*. I HTTP_WebDAV_Server klassen, altså base klassen, er der lavet et hack der opretter en "data" key på options array'et. Denne værdien for denne key er den data der skal sendes/puttes på serveren. Se evt. afsnittet: "HTTP_WebDAV_Server Hacks" senere.

DELETE($options)

Sletter fil eller kollektion (mappe/katalog). Igen parses værdierne frem og tilbage gennem det reference overførte parameter *options*.

PROPFIND(&$options,&$files)

PROPFIND implementere en funktion til at finde properties/egenskaber for filer og kollektioner. Hvis man anmoder om en kollektion, vil den typisk også returnere, de filer som kollektionen indeholder. I modsætningen til det "direktory index" som GET kan printe, så bliver disse properties/egenskaber overført med det reference overførte parameter *options*. Hvorefter base klassen (HTTP_WebDAV_Server) vil parse det til korrekt xml og sende det tilbage som et xml/WebDAV protokol svar og ikke HTML svar.

Hjælp funktioner

Disse funktioner bliver ikke kaldt af base klassen, men af de funktioner vi benytter til at implementere disse protokol kalds funktioner med.

CheckDir($Path , $SQLReturn = "ownerid")

Undersøger om en kollektion eksistere i databasen, returnere false hvis den ikke findes, ellers returneres den værdi der er givet i SQLReturn parametret. Dette er gjort for at undgå unødvendige kald til MySQL databasen, da sagtens kan undersøge om en kollektion eksister og anmode om dens id på sammen tid.

CheckFile($Path)

Checker om en given fil eksistere, returnere true hvis den eksistere og false hvis den ikke eksitere.

CreateDir($Path)

Opretter en kollektion i databasen, og alle underlæggende kollektioner. Det vil sige at hvis du opretter kollektionen /home/mig/test/ vil både /home/, /mig/ og /test/ blive oprettet, hvis de ikke eksistere. Dette gør funktionen ved at kalde på sig selv, og på den måde skabe en loop der altid går pænt op.

GetParentDir($Path)

Returnere overlæggende kollektion, til en given fil eller kollektion. Denne funktion benytter ikke databasen, men foretage ganske enkelte operationerne med streng behandlings funktioner.

GetMimetype($Path)

Forsøger at gætte mimetype ud fra den fil endelse som filen i parametret *path* har. Se evt. kommentar i kildekoden for referencer til relevante standarter.

DelDir($Path)

Sletter den kollektionen der er angivet i parametret *path*, og alle under kollektioner og deres filer. Denne funktion kalder også sig selv hvorved den skaberen løkke der for ryddet pænt op i databasen.

DelFile($Path)

Sletter en fil fra databasen og den indhold fra tabellen binary.

printtodebug($msg,$description)

Skriver en debug meddelelse til debug.log; denne funktion er meget god til at debugge klassen med, da man kan udskrive variabler, mens en anmodning bliver behandlet. Så man lettere kan finde sine fejl, parametret *msg* er den lange meddelelse og *description* er en kort overskrift/beskrivelse.

HTTP_WebDAV_Server Hacks

HTTP_WebDAV_Server er skrevet til folk der vil implementer en WebDAV server, der benytter filsystemet som datalager. Derfor har der været nogle enkelte steder i base klassen hvor vi har foretage nogle små hacks. Disse er dokumenteret her.

http_PUT()

Denne funktion kalder vores PUT() funktion i vores nedarvede klasse og parsede output data der kommer fra vores klasse. Desværre var det bare sådan at vores PUT() funktion kun skal returnere en fil vi har skrive rettigheder til, hvorefter http_PUT(), så skal skrive data til denne fil. Men dette var ikke nogen god løsning, når vi skal have have vores data ind i en database. Derfor lavede vi et lille hack, som downloader filen til en streng og gemmer den i options array'et under nøglen ”data”. Senere bliver options array'et så overført til vores PUT() funktion som så kan indsætte de modtagne data i databasen.

_unslashify()

Da denne klasse ikke er designet til at give adgang til ”/”, kunne unslashify funktionen heller ikke helt arbejde sammen med denne værdi. Derfor måtte vi også lave et lille hack der.